DESCUBRIMIENTO DE LíNEAS DE PROCESOS

JOCELYN SIMMONDS

Jocelyn Simmonds Profesora Asistente Departamento de Ciencias de la Computación, Universidad de Chile. Doctora en Computer Science, University of Toronto, Canadá; Master of Science en Computer Science, Vrije Universiteit Brussel, Bélgica y École Des Mines de Nantes, Francia. Líneas de investigación: Análisis y Diseño de Software, en especial aplicaciones Web y móviles; Validación y Verificación de Sistemas; Educación Apoyada con Tecnología, y cómo atraer mujeres a Ingeniería y Ciencias.
jsimmond@dcc.uchile.cl

ESTE TRABAJO [10] FUE REALIZADO EN CONJUNTO CON FABIÁN ROJAS BLUM, ALUMNO DE DOCTORADO DEL DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN (DCC) DE LA UNIVERSIDAD DE CHILE, Y CON CECILIA BASTARRICA, PROFESORA DEL DCC DE LA UNIVERSIDAD DE CHILE.

Una familia de procesos de software es el conjunto de todos los procesos que una empresa de desarrollo de software sigue. Una empresa puede mantener una familia predefinida de procesos, eligiendo la variante que mejor se aplica para cada proyecto, pero este enfoque tiene problemas de coevolución. Otra posibilidad es definir una línea de proceso de software (SPrL por su acrónimo en inglés): una línea de productos en el dominio de los procesos de desarrollo de software. Para crear una SPrL, la empresa debe definir los elementos comunes de los procesos que se siguen en el día a día, identificando los elementos variables y estableciendo un mecanismo sistemático para generar un proceso a la medida para cada proyecto. Este enfoque permite adaptar el proceso en forma replicable, pero tiene la desventaja de que es caro formalizar una SPrL, requiriendo personal especializado. Para abordar esta problemática, hemos propuesto el *v-algorithm*, que analiza los registros de eventos de proyectos ejecutados por la empresa para identificar los elementos comunes y variables de los procesos aplicados, descubriendo así una línea de procesos.

INTRODUCCIÓN

Las empresas de desarrollo que cuentan con un proceso definido pueden analizar, monitorear, medir y mejorar sus procesos de desarrollo de software [7]. Mas aún, si formalizan sus procesos, pueden automatizar algunas actividades y así reducir el esfuerzo general de desarrollo. El uso de herramientas además facilita la captura de métricas, que son la base para la mejora dinámica del proceso. Formalizar un proceso de software requiere trabajo, y el proceso resultante no siempre se puede usar tal como está definido, dado que los proyectos que desarrolla una empresa tienen diferentes características (proyecto pequeño versus grande, simple versus complejo, uso de tecnología conocida versus desconocida, etc.). O sea, el proceso formalizado debe ser "adaptado" antes de que pueda ser utilizado para guiar un nuevo proyecto [1].

El conjunto de los procesos seguidos por una empresa es una familia de procesos de software. Las empresas pueden mantener una familia de procesos predefinidos, eligiendo para cada proyecto el proceso que mejor se ajusta al proyecto a desarrollar. Este enfoque tiene problemas de coevolución [3]. Por ejemplo, si en un proceso de la familia cambia la forma en que se toman requerimientos, ¿entonces esta actividad debe cambiar en todos los procesos de la familia? Otra opción es definir una línea de proceso de software (SPrL) donde se define un proceso base común con ciertos puntos de variabilidad, como también un mecanimo sistemático para generar un proceso a la medida para cada proyecto [6, 9].

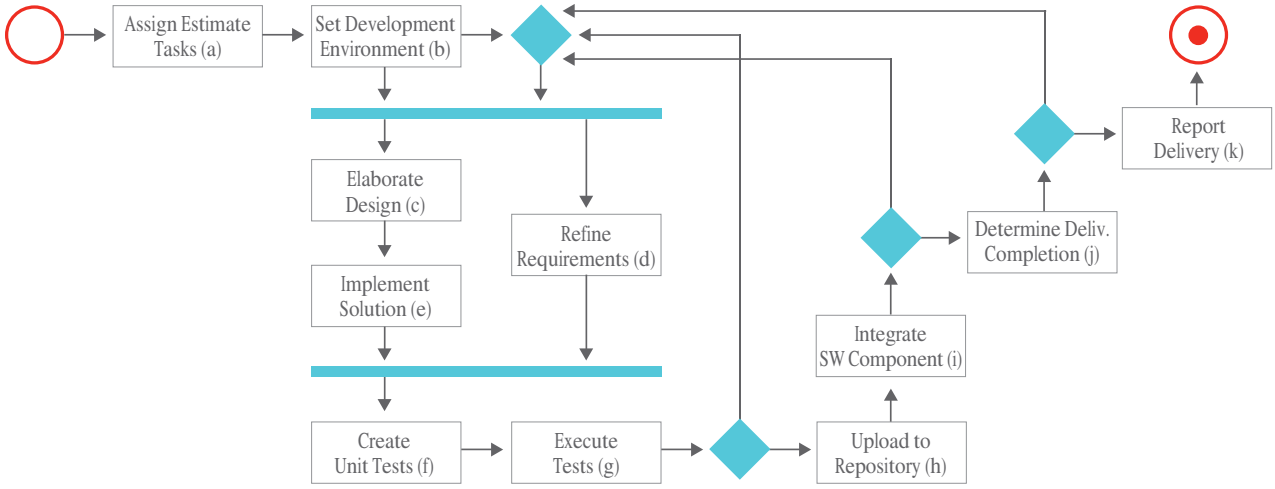

FIGURA 1.
DIAGRAMA DE ACTIVIDAD DEL PROCESO DE DESARROLLO DE MOBIUS (PARCIAL).

En los proyectos Fondef ADAPTE y GEMS hemos seguido un enfoque en base a modelos (MDE por su acrónimo en inglés) para la definición de SPrLs [4], lo que permite automatizar el proceso de adaptación. Como parte de estos proyectos, hemos ayudado a siete pequeñas y medianas empresas chilenas a formalizar sus SPrLs: Rhiscom, KI, Amisoft, Mobius, PowerData, DTS, e Imagen. Formalizar una SPrL no es tarea fácil: solo una de las empresas pudo definir su SPrL en forma independiente. También hemos determinado que los procesos formalizados tienen pobre adopción entre los desarrolladores de la mayoría de estas empresas [2], a pesar de que los ingenieros de proceso se esforzaron para representar los procesos que realmente se aplican. El problema es que muchas veces se modela el proceso que se quiere seguir, y no el que se sigue en la práctica.

DESCUBRIENDO PROCESOS

Éste es uno de los problemas que trata de resolver el área de investigación de Minería de Procesos [12]. Las técnicas de descubrimiento de procesos intentan construir de forma automática un modelo de proceso que describe el comportamiento que se observa en un registro de eventos. En el caso de procesos de software, este registro contiene información acerca del inicio y fin de las distintas actividades de desarrollo. La Figura 1 muestra una parte del proceso de desarrollo de la empresa Mobius, y en el Cuadro 1 se muestra el resumen de un registro de eventos de este proceso para varios proyectos, donde se indica la frecuencia con que ocurre cada traza en el registro de eventos.

Id	Traza	Frecuencia
1	a, b, c, d, e, f, g, h, j, k	10
2	a, b, c, e, d, f, g, h, j, k	10
3	a, b, d, c, e, f, n, h, i, j, k	8
4	a, b, l, d, e, f, g, h, i, j, k	5
5	a, b, d, c, e, m, g, h, j, k	4
6	a, b, d, l, e, m, g, h, j, k	3
7	a, b, d, l, e, f, n, h, j, k	3
8	a, b, l, e, d, m, n, c, e, d, m, n, k	3
9	a, b, c, e, d, f, g, h, o, j, k	2
10	a, b, c, d, f, n, h, k	1
11	a, b, c, e, d, f, k	1

CUADRO 1.
REGISTRO DE EVENTOS DEL PROCESO DE DESARROLLO DE MOBIUS (PARCIAL).

Para extraer un modelo de proceso de un registro de eventos, se deben analizar las relaciones de orden entre las actividades que aparecen en el registro. Por ejemplo, α-algorithm [11] distingue cuatro posibles relaciones de orden entre dos actividades de un registro de eventos: 1) relación directa [$a \succ_L b$]; 2) la causalidad [$a \rightarrow_L b$]; 3) posible concurrencia [$a \|_L b$]; y 4) sin relación directa [$a \nparallel_L b$]. Hay una relación directa entre *a* y *b* si *a* precede *b* en forma directa en alguna traza. Existe una relación de causalidad cuando hay una relación directa entre *a* y *b*, pero no entre *b* y *a*. Dos actividades son posiblemente concurrentes si hay una relación directa entre *a* y *b*, y viceversa. Por último, si no existe una relación directa entre *a* y *b*, y viceversa, se dice que estas actividades no están directamente relacionadas. Sólo una de estas relaciones es verdadera para cada par de actividades.

A partir de esta información se genera una *red de Petri* [8] que representa el comportamiento observado, analizando las relaciones para descubrir patrones de comportamiento. Por ejemplo, si 1) $b \rightarrow_L c$, 2) $b \rightarrow_L d$ y 3) $c \nparallel_L d$, entonces, *c* o *d* se pueden ejecutar después de *b* (patrón XOR split). La Figura 2 muestra una red de Petri generada por α-algorithm para nuestro registro de eventos. Esta red tiene dos tipos de nodos: lugares y transiciones, representados con círculos y rectángulos, respectivamente. En nuestro caso,

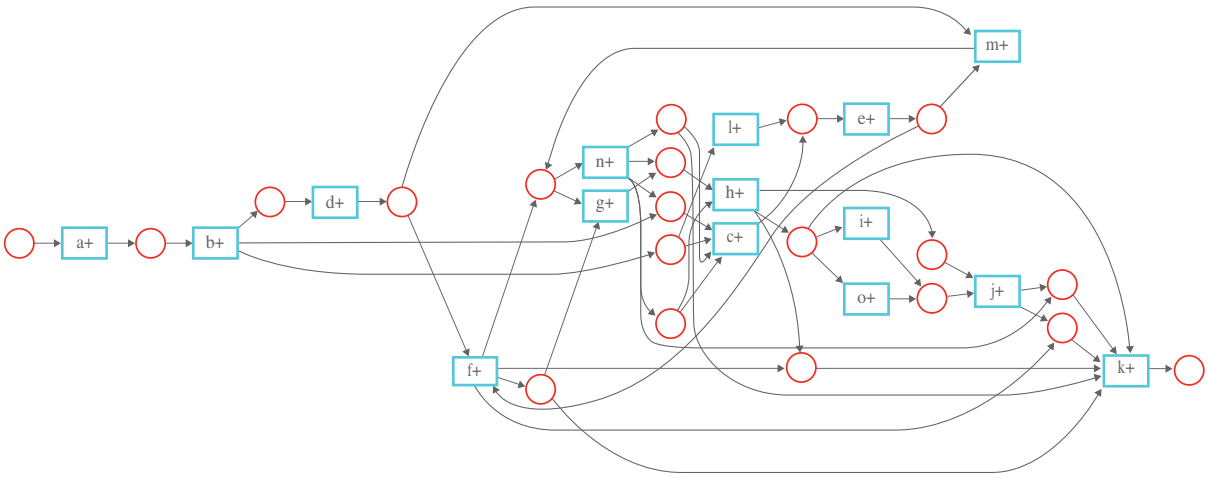

FIGURA 2.
RED DE PETRI GENERADA USANDO α-ALGORITHM.

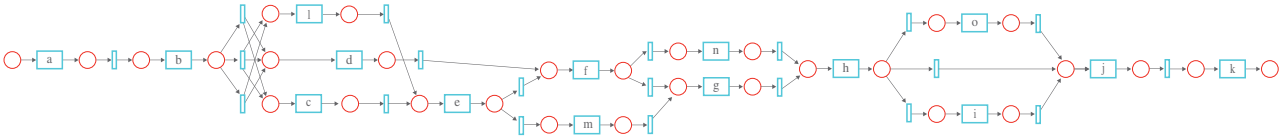

FIGURA 3.
RED DE PETRI GENERADA USANDO HEURISTIC MINER.

las transiciones representan actividades del proceso. Los arcos conectan actividades con lugares y viceversa, pero no nodos del mismo tipo. El proceso resultante es bastante complejo: esto se debe a las pequeñas diferencias que hay entre las trazas del registro de eventos.

Podemos obtener un modelo más simple introduciendo el concepto de "ruido": en la Minería de Procesos, se define como ruido a las trazas excepcionales o de baja frecuencia que ocurren en un registro de eventos.

Por ejemplo, la técnica *Heuristic Miner* [13] le asigna pesos a las actividades, calculando una métrica que indica el grado de certeza de que exista una relación de dependencia entre dos actividades. Al construir el modelo del proceso, esta técnica descarta las dependencias que no cumplen con el umbral de corte fijado por el usuario. La Figura 3 muestra una red de Petri generada con esta técnica para nuestro registro de eventos. Este modelo es más simple que el generado con α-algorithm. El problema es que esto es a costa de ignorar ciertos comportamientos que pueden ser importantes en un proceso. Además, al extraer un solo modelo del proceso, tampoco se pueden identificar las partes de éste que varían dependiendo de las características del proyecto a desarrollar.

DESCUBRIENDO LÍNEAS DE PROCESOS

Obtenemos un proceso más simple al descartar el ruido, pero este modelo es incompleto al ignorar comportamientos poco frecuentes o excepcionales. Podemos mitigar este problema descubriendo una línea de procesos en vez de un solo proceso. Con este fin hemos propuesto el v-algorithm. Este algoritmo construye un modelo del proceso general de desarrollo usando el comportamiento común y frecuente del registro de eventos, y el comportamiento un poco menos frecuente se especifica usando un modelo de variabilidad, el que está relacionado con el proceso general mediante puntos de variación.

Para lograr esto, introducimos dos umbrales (t_1 y t_2) que se usan para dividir un registro de eventos en tres grupos basados en la frecuencia de las relaciones entre actividades. El primer grupo ($f > t_1$) representa el comportamiento común y frecuente que se usa para descubrir el modelo general del proceso. El segundo grupo ($t_1 \geq f \geq t_2$) representa comportamiento poco frecuente pero relevante, por lo que debe ser incluido en el modelo de proceso. Sin embargo, esto haría que el modelo se hiciera mas complejo, así que se especifica en forma separada, como un modelo de variabilidad. El último grupo ($f < t_2$) representa el comportamiento incorrecto o indeseable, o sea, lo que realmente es ruido.

El Listado 1 muestra los pasos principales del v-algorithm:

```
     input : Event log L, t1, t2
     output: SPrL_proc, SPrL_var
1    M,W ← empty matrix
2    frequent, variable ← empty list
3    foreach pair of activities (x,y) in L do
4        M[x][y] ← TypeRelation(x, y)
5        W[x][y] ← Count(x ≻ y)
6        if W[x][y] ≥ t1 then
7            frequent.add((x, y))
8        else if W[x][y] ≥ t2 then
9            variable.add ((x, y))
10   end
     /* Generate base process model        */
11   SPrL_proc ← AlphaAlgorithm(frequent)
     /* Generate variability model         */
12   SPrL_var ← IdOptElements (variable)
13   SPrL_var ← IdAltElements (variable)
```

LISTADO 1.
V-ALGORITHM.

Paso 1: determinar qué relación de orden se cumple para cada par de actividades del registro de eventos, y calcular la frecuencia de estas relaciones. Esta información se guarda en dos matrices: la matriz de relaciones de orden *M*, y la matriz de pesos de las relaciones *W*. En las líneas 1-2 se inicializa *M* y *W*, como también dos listas para guardar las relaciones Frecuentes (*frecuent)* y Variables (*variable).* En las líneas 4-5 se calculan los elementos de *M* y *W*.

Paso 2: los umbrales t_1 y t_2 se usan para generar los grupos de relaciones frecuentes y variables, usando los valores guardados en la matriz *W*. Notar que las relaciones poco frecuentes (ruido) se descartan. En las líneas 6-9 se usan los umbrales t_1 y t_2 para clasificar las relaciones como frecuentes o variables.

Paso 3: una vez clasificadas las relaciones del registro de eventos, se aplica algún algoritmo de descubrimiento de procesos para generar el proceso base del SPrL usando las relaciones frecuentes. En este caso, hemos usado el α-algorithm (ver línea 11). Por último, se construye el modelo de variabilidad usando las relaciones variables (ver líneas 12-13).

Una actividad es opcional en el modelo de variabilidad si: 1) aparece en la lista de relaciones variables, y 2) hay un camino alternativo que evita la ejecución de esta actividad. Una actividad es una alternativa de otra actividad si: 1) aparece en la lista de relaciones variables, y 2) exhibe el mismo comportamiento que otra actividad. O sea, tiene las mismas actividades de entrada y de salida que otra actividad. La Figura 4 muestra la SPrL que descubrimos usando los umbrales $t_1 = 10$ y $t_2 = 4$. Aquí, mostramos el proceso general junto con la información de variabilidad: la actividad *l* (*m*) es una alternativa a la actividad *c* (*f*).

SELECCIÓN DE LOS UMBRALES

Elegimos el valor $t_1 = 10$ porque la traza más frecuente aparece 10 veces en el registro de eventos. Así podemos garantizar que la traza más frecuente será incluida en el proceso base de la SPrL. El valor de t_2 fue elegido para que algunas trazas fueran consideradas como ruido.

Eligiendo distintos valores umbral, se pueden extraer varios SPrL del mismo registro de eventos: ¿cómo podemos decidir cuál de estos SPrL es el mejor? Si dejamos fijo t_2 e incrementamos el valor de t_1, el proceso base se hace más simple porque más comportamiento es considerado como variable. Lo mismo ocurre si dejamos fijo t_1 y disminuimos el valor de t_2.

El ingeniero de proceso debe analizar los modelos generados con distintos umbrales para determinar cuál es el que mejor se ajusta a las necesidades de la empresa. El Grupo de Trabajo de la IEEE sobre Minería de Procesos produjo un manifiesto [5] que identifica cuatro dimensiones no-ortogonales para medir la calidad de un proceso descubierto: ejecutabilidad (*fitness*), simplicidad, precisión, y generalización.

Ejecutabilidad: ¿Cuánto del comportamiento del registro de eventos puede ser ejecutado por el modelo descubierto? Un modelo tiene buena ejecutabilidad si permite la ejecución de gran parte del comportamiento observado en el registro de eventos.

Simplicidad: ¿Cuán complejo es el modelo descubierto? Los modelos de proceso deben ser interpretados por personas, por lo que el modelo más simple que puede explicar el comportamiento observado en el registro es el mejor modelo de acuerdo con esta dimensión.

Precisión: ¿Qué fracción del comportamiento modelado no se observó en el registro de eventos? Un modelo con una buena precisión no permite un comportamiento que es muy diferente del observado en el registro.

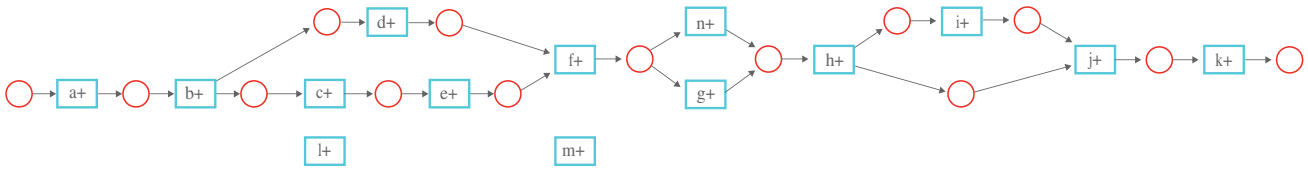

FIGURA 4.
SPrL DESCUBIERTO USANDO V-ALGORITHM ($t_1 = 10$ y $t_2 = 4$).

Generalización: ¿El modelo puede explicar el comportamiento observado en otros registros de eventos del mismo proceso? Como los registros son por definición incompletos, queremos un modelo general que pueda explicar otros registros de un mismo proceso.

Actualmente, estamos estudiando cómo la selección de umbrales afectan estas dimensiones de calidad.

¿EL RUIDO ES REALMENTE RUIDO?

Queremos descubrir un modelo de proceso con el fin de analizarlo y mejorarlo, de modo que se pueda utilizar como un proceso prescriptivo en futuros proyectos [7]. Aquí es correcto eliminar el ruido cuando corresponde a un comportamiento anómalo producido por alguna situación poco frecuente. Por ejemplo, puede ocurrir que el cliente se declaró en quiebra y el proyecto se suspendió. Sin embargo, algunas trazas poco frecuentes pueden ser indicativas de cambios en la empresa. Este es el caso en el que se introducen nuevas formas de abordar ciertas actividades y la empresa quiere adoptar estas prácticas para futuros proyectos. Aquí no se puede descartar el ruido así no más, el ingeniero de proceso debe analizar estas trazas una a una, decidiendo cuáles incluir en el SPrL.

CONCLUSIONES

PARA CONCLUIR, LAS EMPRESAS QUE ESPECIFICAN UN SPrL TIENEN A SU DISPOSICIÓN UN PROCESO RIGUROSAMENTE DEFINIDO, QUE PUEDE SER TAMBIÉN ADAPTADO A LAS NECESIDADES DE CADA PROYECTO DE FORMA ESTRUCTURADA. SIN EMBARGO, REQUIERE BASTANTE TRABAJO ESPECIFICAR UN SPrL, Y NO HAY GARANTÍA DE QUE EL PROCESO MODELADO REPRESENTE EL PROCESO QUE SE LLEVA A CABO EN LA PRÁCTICA. LAS EMPRESAS QUE REGISTREN EVENTOS DE SUS PROYECTOS SE PUEDEN BENEFICIAR DE LOS ALGORITMOS DE DESCUBRIMIENTO DE PROCESO PARA ESTABLECER UN SPrL. CON ESTE FIN SE PROPONE EL V-ALGORITHM, QUE PERMITE DESCUBRIR UNA LÍNEA DE PROCESOS EN VEZ DE UN SOLO PROCESO. ESTO NOS PERMITE LEVANTAR RÁPIDAMENTE UNA LÍNEA DE PROCESOS EN EMPRESAS QUE NO HAN DEFINIDO FORMALMENTE SUS PROCESOS, SIEMPRE Y CUANDO LLEVEN UN REGISTRO DE LOS EVENTOS DE SUS PROYECTOS. LOS RESULTADOS DEL ESTUDIO DE CASO EN MOBIUS SON BASTANTE PROMETEDORES, Y HOY SEGUIMOS TRABAJANDO EN LOS DESAFÍOS QUE HEMOS IDENTIFICADO EN EL PROCESO DE DESCUBRIMIENTO DE LÍNEAS DE PROCESOS. ■

REFERENCIAS

[1] O. Armbrust and H. D. Rombach. The right process for each context: objective evidence needed. In *Proc. of ICSSP'2011*, pages 237241. ACM, 2011.

[2] M. C. Bastarrica, G. Matturro, R. Robbes, L. Silvestre, and R. Vidal. How does Quality of Formalized Software Processes Affect Adoption? In *Proc. of CAiSE'2014*, volume 8484 of *LNCS*, pages 226-240. Springer, 2014.

[3] M. C. Bastarrica, J. Simmonds, and L. Silvestre. Using megamodeling to improve industrial adoption of complex MDE solutions. In *Proc. of MiSE'2014*, pages 31-36, 2014.

[4] J. A. Hurtado Alegría, M. C. Bastarrica, A. Quispe, and S. F. Ochoa. MDE-based process tailoring strategy. *Journal of Software: Evolution and Process*, 26(4):386-403, 2014.

[5] IEEE Task Force on Process Mining. Process-Mining Manifesto. In F. Daniel, K. Barkaoui, and S. Dustdar, editors, *BPM Workshops*, volume 99 of *Lecture Notes in Business Information Processing*, pages 169-194. Springer-Verlag, Berlin, 2012.

[6] T. Martínez-Ruiz, F. García, M. Piattini, and J. Münch. Modelling software process variability: an empirical study. *IET Software*, 5(2):172-187, 2011.

[7] J. Münch, O. Armbrust, M. Kowalczyk, and M. Soto. *Software Process Definition and Management*. Springer, 2012.

[8] T. Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4): 541-580, 1989.

[9] R. Pillat, T. Oliveira, P. Alencar, and D. Cowan. BPMNt: A BPMN extension for specifying software process tailoring. *Information and Software Technology*, 57:95-115, 2015.

[10] F. Rojas Blum, J. Simmonds, and M. C. Bastarrica. Software Process Line Discovery. In *Proc. of ICSSP'2015*, pages 127-136. ACM, 2015.

[11] W. van der Aalst, B. van Dongen, J. Herbst, L. Maruster, G. Schimm, and A. Weijters. Workow mining: A survey of issues and approaches. *Data & knowledge engineering*, 47(2):237-267, 2003.

[12] W. P. van der Aalst. *Process Mining: Discovery, Conformance and Enhancement of Business Processes*. Springer Publishing Company, Inc., 1st edition, 2011.

[13] A. Weijters, W. P. van der Aalst, and A. De Medeiros. Process mining with the heuristics miner-algorithm. *Technische Universiteit Eindhoven, Tech. Rep. WP*, 166:134, 2006.